Amendments to the Specification:

Please replace the two paragraphs beginning on line 10 of page 5 with the following amended paragraphs:

FIG. 6 illustrates an example of a structure of a context [[604]] at design time and at runtime.

FIG. 7 illustrates the context [[604]] at runtime as a set of data instances.

Please replace the paragraph beginning on line 17 of page 5 with the following amended paragraph:

FIG. 1 is a block diagram of an environment for developing an application program 100 using reusable components. The development environment includes an application development framework 105 and a component repository ~~115~~ 110. The application program 100 is developed using reusable components available in the component repository 110, e.g., components 115, 120, and 125. A component in the component repository 110 can have more than one instance, where the component instances are being used in multiple application programs. The application program 100 is developed at design time using the application development framework 105.

Please replace the two paragraphs beginning on line 8 of page 6 with the following amended paragraphs:

The component 200 provides three separate interfaces – a programming interface 205, a data binding interface 210, and a visual interface 215. The programming interface 205 is used by the component embedder (by controller usage 315) to interact with the component 200. The component interface is an active component. The component interface is not just a signature. The component interface defines the component methods that are visible to the component embedder and routes the visible method calls to one or more component implementations. The component embedder embeds the component 200 by programming to the programming interface 205, i.e., the component embedder can call the methods provided by the programming interface 205. In one implementation, the programming interface 205 can be provided by a controller, referred to as a component interface controller. Thus a component embedder can interact with an embedded component through the interface controller of the embedded component.

The component 200 also has one or more visual representations (which will be referred to as views). As described below, a component embedder can access and use the visual representations of the component 200 (for example, to form its own visual representations <u>in view composition 325</u>) through a visual interface 215.

Please replace the two paragraph beginning on line 5 of page 7 with the following amended paragraphs:

FIG. 2B [[is]] illustrates further features of a component 200. The programming interface 205 for the component 200 includes an interface controller 220 and a configuration controller 230. The interface controller 220 implements methods that can be used (e.g., by a component embedder) to interact with the component 200. The configuration controller 230 provides access to configuration data for the component 200. The interface controller 220 has an associated interface context 225 that stores data and state for the interface controller 220. The configuration controller 230 has an associated configuration context 235 that stores configuration data for component 200. The component embedder uses the data binding interface 210 to exchange data with the interface context 225 and the configuration context 235. The runtime framework [[105]] initializes the configuration context 235 when an instance of the component 200 is created at runtime using configuration data provided by the component embedder. The configuration data stored in the configuration context can include data used by the component embedder to customize the component 200, e.g., font size, and selection of fields for a table view.

FIG. 4 is a block diagram of a view. A visual interface of a software application is made up of one or more views arranged in a specific layout. A view 400 specifies a layout of at least one user interface element (UI) element 405, and a view area. UI elements <u>405, 410, 415</u> in a view can include buttons, labels, and menus. The view area defines the area to be occupied by the view 400 in a visual interface embedding the view 400. The UI elements included in the view 400 can include Input UI elements, View UI elements, and Container UI elements. An Input UI element is used to receive input from the user, e.g., a drop down menu, an input field, or a table UI element. A View UI element is used to display application data, e.g., an image view, a text view, or a caption. A Container UI element, described below, is used to include other views and UI elements, e.g., a scroll container UI element having a scroll bar, or a container UI element specifying a layout for included views.

Please replace the paragraph beginning on line 21 of page 9 with the following amended paragraph:

A component developer designates one of the views in the view composition <u>245</u> of the component as an interface view 240. The interface view 240, and the associated inbound plug and outbound plug, are the visual interface for the component 200. At design time, the component embedder can use navigation links to specify view transitions to the interface views 240 of embedded components 200 like any other view in the view composition of the component embedder. A component can present more than one visual interface by defining more than one interface view.

Please replace the paragraph beginning on line 5 of page 10 with the following amended paragraph:

Referring to FIG. 2 the component 200 can also include a component controller 250 that implements common functionality required by views implemented by the component. The component controller receives control when the component is instantiated, after the component instance has been initialized. <u>The component controller 250 has an associated component context 255 for storing data and state for the component controller.</u> The component 200 can also include one or more custom controllers 260, and associated contexts 265. The custom controllers 260 and associated contexts 265 are used to implement and structure functionality and data storage for the component 200.

Please replace the paragraph beginning on line 17 of page 10 with the following amended paragraph:

FIG. 3 is a block diagram of a component embedder 310 using an instance 300 of an embedded component 200, at runtime. The embedded component instance 300 is created at runtime. The embedded component 200 is reusable and several instances 300 of the embedded component 200 can be used at the same time. In the implementation shown in FIG. 3, the runtime framework [[305]] provides the services necessary for managing multiple component instances 300. Services provided by the runtime framework include the creation of component instances, e.g., using a component factory method to create component instances, and managing the lifecycle of component instances, e.g., deleting component instances embedded by a component embedder when the component embedder is deleted. Thus, neither the component embedder nor the embedded component 200 needs to include code for managing multiple component instances 300. Component usage object 305 is an object provided by the application development framework 105 to manage multiple component instances. Each component usage object 305 is associated with a component.

Please replace the paragraph beginning on line 16 page 11 with the following amended paragraph:

The runtime framework [[115]] uses component usage object 305 to access the programming interface of the associated component. The component usage object 305 is also used to manage event subscriptions for the associated component. In an implementation where a component embedder can subscribe to events generated by embedded components, the component usage object 305 caches the event subscriptions for subscribing component, if there is no instance of the subscribing component (because the subscribing component has not been created or because it has been deleted). In such a situation, the event subscriptions are delivered to the subscribing component when the subscribing component is instantiated. Component usage object 305 includes a context mapper 330 that maintains context mappings between the embedder context 320 of the component embedder 310 and the component instance 300. The component usage object 305 caches specified context mappings for a component[[s]] that has not been instantiated, and creates the specified context mappings for the component after the component has been instantiated.